Lab Introduction

This lab focuses on dictionaries, particularly accessing dictionary keys and values. This lab will also give practice on nested data structures, for example a dictionary of dictionaries. Work with a partner to complete the lab activity.

Lab Task

Dictionaries

Like lists, strings, and tuples, dictionaries are a type of collection which can be very helpful in tackling various coding tasks that you may encounter. Similar to how lists are created using square brackets, dictionaries are created using curly brackets {}.

What is different about dictionaries is that they are an unordered collection of elements, made up of what we call “key-value” pairs. Therefore, rather than using numbered indexes to access specific elements in a dictionary, we use the “key” items to access values. Keys may be any immutable data type, such as a string.

For example, if we have a dictionary that keeps track of TAs’ favorite colors, it may look like this:

```
fav_colors = {"Neal": "Purple", "Sai": "Red", "Riley": "Black",
"James": "Green", "Will": "Blue", "Carter": "Light Brown",
"Tino": "Orange", "Rashid": "Pink"}
```

In this dictionary, the TAs’ names are the “keys” and their favorite colors are the values. One key-value pair in this dictionary is “Carter” (key) and “Light Brown” (value).

Looping through Dictionaries

We have several ways to loop through dictionaries, depending on what information we are trying to extract from the dictionary in our loop.

```
for key in fav_colors.keys():
    # This will loop over each key in the dictionary.
    ...

for value in fav_colors.values():
    # This will loop over each value in the dictionary.
    ...
```

```
for key, value in fav_colors.items():
    # This will loop over each item in the dictionary.
    # On each loop, the item tuple will be 'deconstructed' into variables key
and value as expected.
    ...

for item in fav_colors.items():
    # This will loop over each item in the dictionary.
    # The item variable will remain as a tuple: we access the key at index 0,
and value at index 1.
    # The previous method of looping over items is generally preferred for
clarity's sake, but both are equivalent.
    ...

for each in fav_colors:
    # You may be tempted to use this to loop over items...
    # However, it will only loop over the keys!
    # So, we have two equivalent ways to loop over the keys in a dictionary:
    #     for key in fav_colors.keys()
    #     for key in fav_colors
    ...
```

Common Dictionary Operations and Methods

`my_dictionary[a_key] = a_value`

If `a_key` is already a key in the dictionary, this will update the value associated with `a_key`.

If `a_key` is not in the dictionary, this will instead add a new key-value pair to `my_dictionary`.

`my_dictionary[a_key]`

Attempts to return the value associated with `a_key` in the dictionary.

If `a_key` is not found in the dictionary, then this will error with a `KeyError`.

`my_dictionary.get(a_key)`

Attempts to return the value associated with `a_key` in the dictionary.

If `a_key` is not found, by default, it will return `None`, but this value can also be specified like so: `my_dictionary.get(a_key, 1110)` – this will return the integer 1100 if `a_key` is not found.

This method is very useful for when you aren't 100% sure that a given key will be in the dictionary, as it avoids the errors that can arise from using `my_dictionary[a_key]`.

`len(my_dictionary)`

Calculates the number of items, or key-value pairs, in the dictionary.

`del my_dictionary[a_key]`

Removes the key-value pair specified by `a_key` from the dictionary.

If `a_key` is not found in the dictionary, then this will error with a `KeyError`.

`my_dictionary.keys()`

Returns a collection of all the keys in the dictionary.

Most commonly used in a `for` loop.

`my_dictionary.values()`

Returns a collection of all the values in the dictionary.

Most commonly used in a `for` loop.

`my_dictionary.items()`

Returns a collection of all the items in the dictionary, where each item is a tuple containing a key-value pair (in that order).

Most commonly used in a `for` loop.

`a_key in my_dictionary`

Determines whether `a_key` is a key in the dictionary.

Note that this *only* checks for keys. If we need to check whether e.g. a value is in the dictionary, then we'd use `a_value in my_dictionary.values()`.

`my_dictionary.copy()`

Creates a (shallow) copy of the dictionary.

Note that creating a copy of the dictionary is different from saying `new_dictionary = my_dictionary`. If you set a variable equal to a dictionary, then any future edits done on the variable or the dictionary will edit the dictionary stored in both places. Making a copy allows you to preserve the original dictionary while making edits to a new one.

`my_dictionary.pop(a_key)`

Removes and returns the value associated with `a_key`.

By default, if `a_key` is not found in the dictionary, this method will raise a `KeyError`.

However, if we specify a second parameter like so: `my_dictionary.pop(a_key,`

`1110)`, then if `a_key` is not found the integer 1110 will be returned instead without any error..

`my_dictionary.popitem()`
: Removes and returns the item (i.e. a key-value pair, as a tuple) that was last inserted into the dictionary.

Troubleshooting

Remember that with dictionaries, there are no numbered indexes since they are unordered. We can only access values using keys. If you aren't sure what the keys are, you can utilize dict.keys() or dict.items() to iterate through all of them.

Menus

For this lab, you will be utilizing dictionaries to create menus for local Charlottesville restaurants! In addition to creating the menus, you will write 4 functions which allow you to do the following:

1. Add food/drink items to a given menu
2. Calculate how much a specific order from that menu would cost
3. Print out all items in a given menu
4. Calculate how much an order from multiple restaurants would cost

Creating a Menu

First, you will be tasked with initializing a menu that can be used throughout your program. Some examples that you can use are Roots, Ming Dynasty, Ivy Provisions, Al Carbon, Bodo's, or any other Charlottesville restaurant that you like!

1. Create a python file called cville_menus.py
2. Initialize a dictionary with a few key value pairs, where the keys are the names of the menu items (for example, "El Jefe" could be a key for a menu for Roots) and the values are the price of that menu item, as a float (for the El Jefe, that would be 13.25).

For this dictionary, you do NOT need to put the entire restaurant's menu in. 3-5 menu items should be sufficient for now.

Adding Menu Items

Next, you will create a function called add_menu_item that will allow you to add key-value pairs representing menu items to any given menu. This function should have 3 parameters: a dict representing a restaurant's menu, the name of the new menu item, and the price of the menu. Within the function, you should add the new menu item and its price as a key-value pair to the given menu. This function should not print or return anything, it should simply update the menu.

Calculating Order Totals

Next, you will create a function called “calculate_order” which will allow you to see how much a specific order of menu items would cost (including tax and tip). This function should have 3 parameters: a dict of the restaurant’s menu that you’d like to order from, a list that represents all of the items you’d like to order from the restaurant, and an optional argument for tip that is set to 0.18 (18%). For an example of the “order” list, if you wanted to order two El Jefes from Roots, the list would look like this: ["El Jefe", "El Jefe"]

If you try to order something from the restaurant that currently isn’t on the menu, your function should print a message letting the user know that they cannot order that item. If you have extra time at the end of the lab, you can make this more detailed by including an input statement to prompt the user to try and order a different item from the menu.

Once you have gone through the whole order and calculated the total order price, calculate tax and tip. In Charlottesville, the meal’s tax rate is 6.5%. Finally, return the total cost of the order.

Printing the Menu

Next, create a function called print_the_menu that has one parameter, a dict of the restaurant’s menu, and prints out each item on the menu and its corresponding price. The printout of the menu items should be formatted “menu item - price”, each on their own line.

All of Charlottesville Menu

Now we’ll move on to being able to handle multiple Charlottesville restaurant menus at a time! First, you will create a dictionary which houses multiple local Charlottesville restaurants menus Make a new dictionary (call it all_cville_restaurants). The keys will be restaurant names, and the values will be their menus. Put just the one menu that you created at the beginning of the lab in for now Make sample menus for 1-2 other restaurants (try using your add_menu_item function to do so!) Add those menus to the all_cville_restaurants dictionary (where the keys are the restaurant name and the values are the menu)

Ordering for a Party (of picky eaters)
Create a function called place_mega_order(mega_menu, order) that takes in a dict containing restaurant menus (like the one you just made) and a dict representing orders from multiple restaurants, and returns the total cost. For the second parameter that represents orders from multiple restaurants, each key inorder should be a restaurant name, and each corresponding value should be a list of items to order from the restaurant (ex. {"Roots":["El Jefe", "The Mayweather"]}.

You should call your implementation of calculate_order from inside place_mega_order (probably in a loop).

Conceptual Questions

When checking you off, a TA will ask you one of the following questions. Discuss the following questions with your partner as you are working through the lab.

- What is the fundamental difference between dictionaries and other collections like lists?
- Give an example of when to use a dictionary over a list. Give an example of when to use a list over a dictionary.
- How are dictionaries structured and what are the key-value pairs?
- Explain how to access values in a dictionary using keys.
- How do you iterate through a dictionary and what are the different methods available for iteration?
- How do you check if a specific key is present in a dictionary?
- How would you access a specific value within a nested dictionary?